EXPANDED BASIC

KISS GY.
1987.
OK.
25.

A program 30 új szót definiál az eredeti BASIC szavak mellé, és három kihasználatlan funkciót is felhasználttá tesz, valamint a LIST, EDIT és DELETE utasitások "bogaraitt" is kijavitja. A három felhasználttá tett funkció: a MERGE utasitás (leirást ld. a parancsoknál), a TIME$ és az & függvények (leirást ld. a függvényeknél).

A LIST, EDIT és DELETE utasitások megszüntetett "bogarai":

1./ Ha egy sorba macskakörmök közé (a lenyil használatával) 128 feletti kódú karaktereket vittünk be, listázáskor, és editáláskor ezek helyett az ilyen kódú utasitások jelentek meg.

2./ Ha Ø (nulla) sorszámú sort javitottunk, a sorszám nem jelent meg.

3./ Ha editáláskor BRK-et nyomt az editálásból kiléptünk ugyan, de soremelés nem történt a képernyőn.

4./ Ha a DELETE utasitásban nemlétező sorra hivatkoztunk, vagy DELETE utasitást próbáltunk meg használni, hibaüzenetet kaptunk. Ezentúl a DELETE utasitás szintaxisa megegyezik a LIST utasitásával, egyetlen különbség, hogy ha csak DELETE-et irunk nem törlődik az egész program, hanem hibaüzenetet kapunk.

5./ Ha 210 karakternél hosszabb sort javitottunk, a gép nem korlátozta a sorhosszt, és megőrült, jobb esetben elszállt. Most, ha 210 karakternél hosszabb sort javitunk, a gép addig nem enged beszúrást végezni, mig a sorhosszt nem csökkentjük 210 karakter alá.

A program további szolgáltatása, hogy teljes szövegű hibaüzeneteket ad. Ide tartozik még, hogy a tokenizálás módositása miatt szükség volt egy új hibaüzenet bevezetésére:

pore 16 454, 1 gyors forrat

?Line too long error-t kapunk, ha a tokenizálás során a sor hosszabb lett 255 byte-nál (pl.: ha shift + lenyil + csillag karakterrel - LOAD tokennel végig irjuk a sort, majd elküldjük).

A beépitett óra kikapcsolása szükségessé válik bármely file-kezelő utasitás esetén. Ez automatikusan megtörténik, ha ezeket az utasitásokat a programunkba akkor gépeltük, amikor az Expanded BASIC rendszerrel dolgoztunk. De ha egy olyan programot folytatunk, amibe normál BASIC-kel gépeltünk, file-kezelő utasitásokat, és a pgoramukkal az órát is működtetni kivánjuk, jobb ha az összes olyan sort, melyben file-kezelő utasitás van, lehivjuk edit-be, és shift + előrenyillal elküldjük minden különösebb javitás nélkül. (Kivételt képeznek az olyan sorok, melyekben csak PRINT# vagy INPUT# utasitás szerepel.) Ha file kezelő utasitást használunk, és az óra be van kapcsolva az utasitás végrehajtása alatt a kijelzésre, a file visszaolvashatatlan lesz.

Az Expanded BASIC-kel irt (átirt) programokat ne javitsuk és futtassuk a normal BASIC-kel. Ezt, ha speciális utasitás szerepel a programban nem is tudjuk megtenni.

Az Expanded BASIC új szavainak leirása:

Az Expanded BASIC szavai három csoportba oszthatók: parancsok, utasitások, és függvények.
A Parancsok csak parancs módban használhatóak, program módban ? Syntax error-t eredményeznek. Az utasitások és a függvények mind parancs, mind program módban használhatók.

I. Parancsok:

REN a,b (RENUMBER) sorszámozza a programot a-tól kezdve b lépésközzel (az elhagyott paraméter értéke automatikusan 1Ø lesz). Természetesen a programban szereplő GOTO, GOSUB, THEN, ELSE, RUN, RESTORE, RESUME, ERL=, ERL>, ERL<, ERL<>, ERL=>, és ERL<= utasitások mögötti sorszámokat is átsorszámozza.

FIND szöveg Keresés; a BASIC program olyan sorait, melyben előfordul a szöveg, kilistázza a képernyőre. Folyamatos listázáshoz egy billentyűt (kivétel:BRK) folyamatosan nyomnunk kell. A BRK megnyomására a gép visszakerül parancs üzemmódba. Ha egy parancsot akarunk megkeresni, nem elég a parancs egy részét beirni a FIND mögé, hanem a teljes parancsot be kell gépelni. (pl.: ha NEXT utasitásokat akarunk keresni, nem elég azt beirni, hogy FIND NEXT.)

CMCT (COMPACT) tömöriti a BASIC programot. A programban található felesleges kettőspontokat és szóközöket és REM utasitásokat kiszedi a programból, majd 1-től kezdve egyesével átsorszámozza azt. (Természetesen a macskakörmök közötti szóközöket és kettőspontokat nem bántja.) Ezzel a funkcióval programunk hosszát jelentősen csökkenthetjük.

VARS (VARIABLES) Változók. A parancs kilistázza a program változóit, és ezek értékeit a képernyőre. Folyamatos listázáshoz bármely billentyűt (kivétel: BRK) folyamatosan kell érinteni. A BRK megérintésére a gép újra parancs üzemmódba kerül.

ARRS (ARRAYS) Tömbök. Lásd VARS, csak a tömböket listázza.

TYPS (TYPES) Tipusok. A változók alapértelmezésű tipusát kilistázza a képernyőre. (Ezeket a tipusokat tudjuk megváltoztatni a DEFIT, DEFSNG, DEFDBL, DEFSTR utasitásokkal.)

OLD Régi. A NEW utasitással törölt programot visszaállitja, ha időközben még nem vettünk fel változókat.

APPEND "filenév"

Ráakaszt. A memóriában lévő program után tölti a szalagon lévő filenév nevű file-t. (ha nem adunk meg nevet, az első útjába kerülő file-t tölti be.). A nembetöltött blokkokat, melyekben képernyők, vagy assembler programok találhatók, inverz blokkszám kijelzéssel jelzi a gép. Az auto startok éppen ezért nem hatnak.

Ha az eredeti programban szerepel egy 1Ø-es sor, és a kazettán is van egy 1Ø-es sor, a programunk két 1Ø-es sort fog tartalmazni, ami a BASIC megzavarodását okozhatja.(de lásd a MERGE parancsot!)

MERGE "filenév"

EGYBEOLVASZT. A memóriában lévő programhoz hozzáfűzi a szalagon lévő programot, majd az eredetileg a memóriában lévő programot 1-től kezdve egyesével átsorszámozza, ezután a keletkezett utolsó sorszám + 1-től kezdve egyesével átsorszámozza a kazettáról betöltött programot, végül az egészből egy programot csinál. A file betöltésre igaz az APPEND parancs file betöltése.

HELP SEGIT. Az utolsó hibás sort kilistázza, a hibás utasitást inverzen. Ha ez nem történik meg valahogyan töröltük a hiba állapotot.

II. UTASITÁSOK: CLS: PRINT$Ø,2,CHR(8)CHR(4)"␣KISS GY.␣1987.Ø7.", S, 178,191, B, 1, VRUN

WRITE "filenév"...

IR. Tetszőleges file generálható vele a megadott névvel. A file tartalmazhat képernyő darabokat, assembler v. BASIC programot. A file lehet autostartos BASIC v. assembler file is. Az egyes blokkokat külön-külön kell megadni a név után egymástól vesszővel elválasztva őket. Képernyő blokk megadása: S, az elsőnek kimentendő képernyősor száma, az utolsónak kimentendő képernyősor száma. Assembler blokk megadása: A, az elsőnek kimentendő byte cime, ! ahova az elsőnek kimentett byte majd betöltődik, az utolsónak kimentendő byte cime AUTO inditási cim. A második cim elhagyása esetén a file oda töltődik, ahonnan kimentettük. Az AUTO és az utána lévő cim elhagyása esetén a file nem lesz autostartos, és további blokkok adhatók meg. BASIC blokk megadása: B, az elsőnek kimentendő programsor sorszáma, az utolsónak kimentendő programsor sorszáma RUN.

Ha elhagyjuk a RUN-t a program nem lesz autostartos, és további blokkok adhatók meg. Ha BASIC és Assembler programot is kimentünk, és Assembler autostartot rendelünk el, a gépi kódú programunknak gondoskodnia kell a BASIC mutatók állitásáról! (Legegyszerűbb ha igy kezdődik a program: CALL 1AF8H; INC HL; LD (4ØF9H), HL; CALL 1B61H. Ezek után a stack üres!)

CLOCK ON — Óra bekapcsolása. Ezentúl az óra minden másodpercben 1-gyel nő, és kijelzésre kerül a képernyő jobb felső sarkában (ÓÓ:PP:MM formában.)

CLOCK OFF — Órakijelzés letiltása. Az óra eltünik a jobb felső sarokból (ha ott volt, de a gépben jár tovább.

CLOCK STOP — Az óra megállitása. Az óra eltünik a jobb felső sarokból (ha ott volt) és megáll.

CLOCK START — Az óra inditása. Az óra elindul (ha állt, egyébként jár tovább), de nem kerül kijelzésre.

CLOCK SET "ÓÓ:PP:NN" — Az óra beállitása. Az óra a sting kifejezésben megadott értéket veszi fel. (A sting kifejezés formátuma: 8 karakter hosszú, az első két karakter az óra, a harmadik karakter kettőspont, a negyedik és az ötödik a perc, a hatodik szintén kettőspont, a hetedik és a nyolcadik a másodperc. Az óra ØØ-tól 23-ig terjedhet, a perc és a másodperc ØØ-tól 59-ig.

DPOKE cim,...,... — A memória átirása kétbyte-os számokkal (Dauble POKE). A megadott cimre helyezi az utána álló számo(ka)t. A felsorolt számot tetszőleges egészek - 32768-tól 32767-ig. A számok alacsonyabb byte-ja kerül a cimre, majd a cim nő eggyel és ide a felső byte kerül majd megint nő eggyel a cim és ide a következő szám alacsonyabb byte-ja kerül ... egészen addig, mig a listának nincs vége.

SPOKE cim,string,string

A memória átirása a stringekben lévő karakterek értékeivel. (String POKE). A cimtől kezdődően beirja a memóriába a stringekben lévő karakterek kódjait, minden egyes karakter után eggyel növelve a cimet, mig a stringek listájának végére nem ér.

III. FÜGGVÉNYEK:

XOR (A,B) (eXclusive OR) Kizáró vagy függvénykapcsolat (bitenkénti). Az A és B egész kifejezést egymással kizáró vagy logiaki függvénykapcsolatba hozza, és ezt vissza adja eredményül.

DPEEK (cim) (Double PEEK) A cimen lévő byte értékéhez hozzáadja a cim + 1-en lévő byte értékének 256 szorosát.

BIN$ (egész kif.)

A függvény eredménye egy 16 karakter hosszú bináris karaktersorozat, mely csak Ø és 1 karaktereket tartalmaz, és melynek bináris értéke a zárójelek közötti kifejezés értéke. A kifejezés értéke - 32768-tól 32767-ig terjedhet.

BIN (string kif.) A függvény eredménye egy egész szám - 32768 és 32767 között, melynek értéke a zárójelek közötti stringkifejezés bináris értékével egyezik meg.

CFP (egész kif.) (Convert to Floating Point = lebegőpontossá alakitás.) A zárójelek közötti egész kifejezést lebegőpontossá konvertálja, és ha az kisebb volt nullánál, akkor hozzáad 65536-ot, igy a függvény eredménye egy Ø és 65535 közötti egyszeres pontosságú szám.

CTI (kifejezés) (Convert To Integer = egésszé alakitás) Ha a kifejezés string, akkor azt mint hexadecimális karaktersorozatot értékeli ki. Ha normális egyszeres vagy kétszeres pontosságú szám, akkor egésszé alakitja, de ha nagyobb volt a kifejező értéke, mint 32767, akkor levon belőle 65536-ot, igy a függvény értéke - 32768-tól 32767-ig terjedhet. Ha egész számot adunk meg a kifejezésben, a függvényérték ez a szám lesz.

ADDR$ (egész kif.)
(ADDRESS = cim.) A függvényérték egy olyan 4 karakter hosszú string, mely hexadecimálissá alakitva tartalmazza az egész kifejezés értékét. (ami - 32768-tól 32767-ig terjedhet.)

BYTE$ (egybyte-os kif.)
A függvényérték egy olyan 2 karakter hosszú string, mely hexadecimálissá alakitva tartalmazza az egybyte-os kifejezést (melynek értéke Ø-tól 255-ig terjedhet).

HEXA$ (egész kif.) A függvényérték hexadecimálissá alakitva tartalmazza (string formájában) az egész kifejezést (melynek értéke - 32768-tól 32767-ig terjedhet), de úgy, hogy a felesleges nullák nem jelennek meg.

PEEK$ (cim, hossz, maszk)
A függvény egy olyan stringet ad eredményül, melyben a megadott cimtől megadott darabszámú byte van felhozva, egyesével bitenkénti logikai és kapcsolatban hozva a maszk-kal. Ha maszkot nem adunk meg, értéke automatikusan 255 lesz. A hossz értéke 1-től 255-ig terjedhet. Nulla hossz esetén üres stringet kapunk, de a többi stringváltozó értéke megzavarodik.

VALUE (string kif.)

A függvényérték tipusa függ a megadott stringtől. A függvény ugyanis a megadott stringet mint aritmetikai kifejezést tekinti, és kiszámitja a helyettesitési értékét, attól függetleniil, hogy az stringet, vagy számváltozót eredményez-e. Pl.: ha A$ értéke az, hogy "5 * 6" a VALUE (A$) függvény eredménye az, hogy 30, és tipusa számváltozó. Ha A$ értéke az, hogy "CHR$(42)" az eredmény egy karakter hosszú és az, hogy "*". (természetesen ez csak egy stringben tárolható.) Ha a VALUE függvényre ? Syntax error-t kapunk, és a függvény formátuma tökéletes, lehet hogy a független változóban (a megadott stringben) keletkezett a hiba. Pl.: ? VALUE ("5 * (6+z") utasitásra ? Syntax error-t kapunk, mert a stringben egy zárójel nincs bezárva.

TOKEN$ (string kif.)

A megadott stringet tokenizálja, és ezt visszaadja eredményül. A visszaadott stringben a müveleti jelek és függvények, valamint az utasitások helyett tokenjük lesz található..A függvényre azért van szükség, mert ha a VALUE-t sorozatosan ugyanazzal a stringgel használjuk, az minden alkalommal tokenizálja, és ez elég hosszú idő. Tehát: ha egy stringet sokszor kell függvényként végigszámolni, akkor gyorsabb, ha az elején a TOKEN$ függvénnyel tokenizáljuk és utána ezt a stringet számoltatjuk végig.

FUNC (string kif.) A TOKEN$-sal tokenizált stringek helyettesitési értékét számitja ki. (mint a VALUE, csak ennek már tokenizált stringet kell átadni.)

TIME$ A függvény értéke egy 8 karakter hosszú string, melyben az óra aktuális értékét kapjuk vissza ÓÓ:PP:MM formában.

& hexadecimális számsor

Hexadecimális számmegadást tesz lehetővé. Pl. a PRINT & 56CE utasitás hatására megjelenik az